UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/803,659	03/18/2004	Jos Manuel Accapadi	AUS920031016US1	6003

35525 7590 11/26/2008

IBM CORP (YA)
C/O YEE & ASSOCIATES PC
P.O. BOX 802333
DALLAS, TX 75380

EXAMINER	
ZHE, MENG YAO	
ART UNIT	**PAPER NUMBER**
2195	

NOTIFICATION DATE	DELIVERY MODE
11/26/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

ptonotifs@yeeiplaw.com

Office Action Summary	Application No.	Applicant(s)	
	10/803,659	ACCAPADI ET AL.	
	Examiner	Art Unit	
	MENGYAO ZHE	2195	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *27 August 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-20 are presented for examination.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

3. Claims 1-2, 13, 17 are rejected under 35 U.S.C. 102(a) as being anticipated by Brenner et al., Patent No. 6,658,449 (hereafter Brenner).

4. As per claims 1, 17, Brenner teaches a method of queuing threads among a plurality of processors in a multiple processor system having a plurality of multi-processor modules, wherein each of the plurality of multi-processor modules comprises a plurality of processors, wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues (Fig 2; Column 3, lines 1-17), the method comprising the computer implemented steps of:

receiving a first thread to be processed, wherein the first thread belongs to a first process (Column 4, lines 66-67);

identifying whether the first thread is a bound thread associated with a first processor of the plurality of processors (Column 4, lines 10-12);

responsive to identifying that the first thread is the bound thread associated with the first processor, assigning the first thread to a first local run queue of the plurality of local run queues, wherein the first local run queue is associated with the first processor (Column 4, lines 10-16);

responsive to not identifying that the first thread is the bound thread, identifying whether the first thread is part of an existing process on a first multi-processor module of the plurality of multi-processor modules (Column 4, line 66-Column 5, line 6);

responsive to identifying that the first thread is part of the existing process on the first multiprocessor module, attempting to identify an idle processor that has executed a second thread belonging to the first process, wherein the idle processor is identified from the plurality of processors of the first multi-processor module associated with the existing process (Column 4, line 66-Column 5, line 6);

responsive to identifying that the idle processor that has executed the second thread belongs to the first process, assigning the first thread to a second local run queue of the plurality of local run queues, wherein the second local run queues is associated with the idle processor (Column 4, line 66-Column 5, line 6; Column 5, lines 7-15)

responsive to not identifying that the idle processor that has executed the second thread belonging to the first process, assigning the first thread to a first chip run queue,

wherein the first chip run queue is associated with the first multi-processor module (Column 5, lines 16-26).

5. As per claims 2, 13, Brenner teaches assigning the first thread the chip run queue dedicated to the first multi-processor module (Column 5, lines 16-26).

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1, 2, 13, 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sydir et al., Pub 2005/0141715 (hereafter Sydir) in view of Zolnowsky, Patent No. 5,826,081 (hereafter Zolnowsky).

8. As per claims 1, 17, Sydir teaches a method of queuing threads among a plurality of processors in a multiple processor system having a plurality of multi-processor modules, wherein each of the plurality of multi-processor modules comprises a plurality of processors, wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip run queues, and wherein each of the plurality

of processors is associated with one of a plurality of local run queues (Fig 1; Para 20, 21), the method comprising the computer implemented steps of:

receiving a first thread to be processed; identifying the first thread as part of an existing process on a first multi-processor module of the plurality of multi-processor modules (Para 35);

attempting to identify an available processor that has executed a second thread belonging to the first process, wherein the available processor is identified from the plurality of processors of the first multi-processor module associated with the existing process (Para 38)

responsive to identifying that the idle processor that has executed the second thread belongs to the first process, assigning the first thread to a second local run queue of the plurality of local run queues, wherein the second local run queues is associated with the available processor (Para 39, 40)

responsive to not identifying that the available processor that has executed the second thread belonging to the first process, assigning the first thread to a first chip run queue, wherein the first chip run queue is associated with the first multi-processor module (Para 42).

Sydir does not specifically teach determining whether the thread is bound or unbound and that the available processor is actually an idle processor.

However, Sydir does teach that in the special case when all cores are idle, then the thread may be assigned to the idle core for the purpose of assigning the thread to a processor regardless (Para 44).

It would have been obvious to one having ordinary skill in the art at the time of the applicant's invention to have the thread be actually assigned to an idle processor that has specifically executed the second thread belonging to the first process, because it allows for the thread to run regardless.

Zolnowsky teaches assignment of bounded and unbounded threads including identifying whether the first thread is a bound thread associated with a first processor of the plurality of processors (Column 7, lines 1-6; Column 10, lines 5-10); and responsive to identifying that the first thread is the bound thread associated with the first processor, assigning the first thread to a first local run queue of the plurality of local run queues, wherein the first local run queue is associated with the first processor (Column 7, lines 1-6) for the purpose of determining a set of allowable processor for a thread to run on.

It would have been obvious to one having ordinary skill in the art to combine the teachings of Sydir with the specifics of assignment of bounded and unbounded threads including identifying whether the first thread is a bound thread associated with a first processor of the plurality of processors and responsive to identifying that the first thread is the bound thread associated with the first processor, assigning the first thread to a first local run queue of the plurality of local run queues, wherein the first local run queue

is associated with the first processor, as taught by Zolonowsky, because it helps to determine a set of allowable processor for a thread to run on.

9. As per claims 2, 13, Sydir teaches assigning the first thread the chip run queue dedicated to the first multi-processor module (Para 42).

10. Claims 1-6, 13-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Alfieri, Parent No 5,745,778 (hereafter Alfieri) in view of Zolnowsky, Patent No. 5,826,081 (hereafter Zolnowsky) further in view of Kimmel et al., Patent No. 6,105,053 (hereafter Kimmel).

11. Alfieri and Kimmel were cited in the previous office action.

12. As per claims 1, 17, Alfieri teaches a method of queuing threads among a plurality of processors in a multiple processor system having a plurality of multi-processor modules, wherein each of the plurality of multi-processor modules comprises a plurality of processors (Fig 1: units 100-103 form one processor module and units 104-107 form another processor module), wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues (Column 9, lines 40-52: level 1 queue would correspond to the chip run queue and level 0 queues for each processor corresponds to local run queues), the method comprising the computer implemented steps of:

receiving a first thread to be processed; identifying the first thread as part of an existing process on a first multi-processor module of the plurality of multi-processor modules (Column 6, lines 31-36; Column 9, lines 55-57; Column 10, lines 1-3);

attempting to identify an available processor that has executed a second thread belonging to the first process, wherein the available processor is identified from the plurality of processors of the first multi-processor module associated with the existing process (Column 6, lines 31-36; Column 9, lines 58-61)

responsive to identifying that the idle processor that has executed the second thread belongs to the first process, assigning the first thread to a second local run queue of the plurality of local run queues, wherein the second local run queues is associated with the available processor (Column 7, lines 34-40)

responsive to not identifying that the available processor that has executed the second thread belonging to the first process, assigning the first thread to a first chip run queue, wherein the first chip run queue is associated with the first multi-processor module (Column 7, lines 40-44).

Alfieri does not specifically teach determining whether the thread is bound or unbound and that the available processor is actually an idle processor.

However, Zolnowsky teaches assignment of bounded and unbounded threads including identifying whether the first thread is a bound thread associated with a first processor of the plurality of processors (Column 7, lines 1-6; Column 10, lines 5-10); and responsive to identifying that the first thread is the bound thread associated with the

first processor, assigning the first thread to a first local run queue of the plurality of local run queues, wherein the first local run queue is associated with the first processor (Column 7, lines 1-6) for the purpose of determining a set of allowable processor for a thread to run on.

It would have been obvious to one having ordinary skill in the art to combine the teachings of Alfieri with the specifics of assignment of bounded and unbounded threads including identifying whether the first thread is a bound thread associated with a first processor of the plurality of processors and responsive to identifying that the first thread is the bound thread associated with the first processor, assigning the first thread to a first local run queue of the plurality of local run queues, wherein the first local run queue is associated with the first processor, as taught by Zolnowsky, because it helps to determine a set of allowable processor for a thread to run on.

Kimmel teaches that the available processors may actually be idle processors for the purpose of load balancing (Column 10, lines 60-65).

It would have been obvious to one having ordinary skill in the art at the time of the applicant's invention to modify the teachings of Alfieri in view of Zolnowsky with threads being assigned to idle processors, as taught by Kimmel, because it allows for load balancing.

13. As per claims 2, 13, Alfieri teaches assigning the first thread the chip run queue dedicated to the first multi-processor module (Column 9, lines 53-57).

14. As per claim 3, Alfieri teaches identifying the first multi-processor module as associated with the existing process (Column 6, lines 31-36; Column 9, lines 55-57; Column 10, lines 1-3).

15. As per claim 4, Alfieri teaches being able to identify threads having the same processor and multi-processor module (Column 6, lines 31-36; Column 9, lines 55-57; Column 10, lines 1-3).

But Alfieri does not specifically teaches wherein the step of identifying the first multi-processor module further comprises: maintaining a record of processes having threads executed by a processor of the first multi-processor module during a predetermined preceding interval.

However, it would have been obvious to one having ordinary skill in the art at the time of the applicant's invention to see that in order for Alfieri's invention to keep track which processor must be used to execute an application, there must be a record that records processes executed by a processor.

16. As per claim 5, Alfieri teaches identifying one of the plurality of processors of the first multi-processor module as an idle processor; and assigning the first thread to a

local run queue of the plurality of local run queues associated with the idle processor (Column 6, lines 31-36; Column 9, lines 58-61).

17. As per claim 6, Alfieri teaches wherein the step of identifying further comprises: reading attribute information of the first thread (Column 6, lines 31-36; Column 9, lines 55-57; Column 10, lines 1-3).

18. As per claim 14, Alfieri teaches third instructions for identifying a process associated with the first thread (Column 3, lines 1-5, lines 29-51), wherein the second instructions identify threads of the process assigned to the first multi-processor module (Column 6, lines 31-36).

19. As per claim 15, Alfieri teaches third instructions for comparing a thread load of the first queue with a thread load of a second queue dedicated to a second multi-processor module of the plurality of multi-processor modules; and fourth instructions for reassigning the first thread to the second queue (Column 8, lines 61-63).

20. As per claim 16, Alfieri teaches third instructions for reassigning the first thread to a second queue dedicated to a processor of a second multi-processor module of the plurality of multi-processor modules (Column 9, lines 52-57).

21. As per claim 18, Kimmel teaches wherein the first multi-processor module comprises a plurality of central processing units disposed on a first chip, and the second multi-processor module comprises a plurality of central processing units disposed on a second chip (Column 4, lines 17-30).

22. As per claim 19, Kimmel teaches wherein the first multi-processor module is a simultaneous multi-threading central processing unit, and the second multi-processor module is a simultaneous multi-threading central processing unit (Column 5, lines 15-20).

23. As per claim 20, Kimmel teaches wherein the scheduler identifies a second thread of a process associated with the first thread, and the second thread is assigned to the chip run queue (Column 2, lines 36-41; Column 11, lines 1-6).

24. Claims 7-12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kimmel et al., Patent No. 6,105,053 (hereafter Kimmel) in view of Sundaresan, Patent No. 6,289,369 (hereafter Sundaresan) further in view of Chen et al., Patent No. 7,062,556 (hereafter Chen) further in view of Zolnowsky, Patent No. 5,826,081 (hereafter Zolnowsky).

25. As per claim 7, Kimmel teaches a method of load balancing threads among processors in a multiple processor system having a plurality of multi-processor modules (Fig 1A, units 10, 11, 12; Fig 1B, units 110, 111), wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues (Column 24, lines 37-55: each node along the tree gets a queue, which includes the root nodes that are the processors), the method comprising the computer implemented steps of:

performing, by an idle processor of the plurality of processors a first multi-processor module of the plurality of multi-processor modules, a first attempt at a thread steal from a first one of the plurality of local run queues of one of the plurality of processors located on the first multi-processor module for reassignment of a thread to a second one of the plurality of local run queue associated with the idle processor (Column 11: lines 21-27: stealing within a module);

responsive to failure of the first attempt, performing a second attempt at a thread steal from one of the plurality of chip run queues associated with a second multi-processor module of the plurality of multi-processor modules and assigning the first thread to either one of the plurality of chip run queues or one of the plurality of local run queues (Column 10, lines 60-63; Column 11, lines 27-40: stealing amongst modules).

Kimmel does not specifically teach responsive to performing the first steal attempt, stealing the thread from the first one of the plurality of local run queues if the first one of the plurality of local run queues has a largest number of threads of all the local run queues of the multi-processor module, if the first one of the plurality of local run queues contains more threads than an intra-module steal threshold of the first multi-processor module, and if the thread is an unbound thread; and responsive to performing the second steal attempt, stealing the thread from the one of the plurality of chip run queues associated with the second multi-processor module of the plurality of multi-processor modules, if the one of the plurality of chip run queues has a largest number of threads of all of the plurality of chip run queues of the multi-processor system, and if the one of the plurality of chip run queues contains more threads than an inter-module steal threshold of the first multi-processor module.

However, Sundaresan teaches stealing the thread from the first one of the plurality of local run queues if the first one of the plurality of local run queues has a largest number of threads of all the local run queues of the multi-processor module for the purpose of load balancing (Column 2, lines 48-51).

It would have been obvious to one having ordinary skill in the art at the time of the applicant's invention to combine the teachings of Kimmel with the specifics of stealing the thread from the first one of the plurality of local run queues if the first one of the plurality of local run queues has a largest number of threads of all the local run queues of the multi-processor module, as taught by Sundaresan, because it allows for load balancing.

Chen teaches stealing the thread from the first one of the plurality of local run queues if the first one of the plurality of local run queues contains more threads than an intra-module steal threshold of the first multi-processor module for the purpose of alleviating loads (Column 6, lines 4-20).

It would have been obvious to one having ordinary skill in the art at the time of the applicant's invention to combine the teachings of Kimmel in view of Sundaresan with stealing the thread from the first one of the plurality of local run queues if the first one of the plurality of local run queues contains more threads than an intra-module steal threshold of the first multi-processor module, as taught by Chen, because it alleviates loads.

Zolnowsky teaches bounded and unbounded threads for the purpose of limiting a set of allowable processor for a thread to run on (Column 7, lines 1-6; Column 10, lines 5-10).

It would have been obvious to one having ordinary skill in the art to combine the teachings of Kimmel in view of Sundaresan further in view of Chen with the specifics of

assignment of bounded and unbounded threads, as taught by Zolonowsky, because it helps to limit a set of allowable processor for a thread to run on.

26. As per claim 8, Kimmel teaches evaluating a criterion associated with the second multi-processor module; and responsive to evaluating the criterion, determining if a thread is to be reassigned from the one of the plurality of chip run queues to the local run queue of the idle processor (Column 11, lines 25-40).

27. As per claims 9, Kimmel teaches reassigning a thread of the one of the plurality of chip run queue to the local run queue of the idle processor (Column 11, lines 1-6).

28. As per claim 10, Kimmel teaches responsive to failure of the second attempt, performing a third attempt at a thread steal from one of the plurality of local run queues associated with one of the plurality of the second multi-processor module for reassignment of a thread to the second one of the plurality of local run queues associated with the idle processor (Column 10, lines 60-63; Column 11, lines 27-40).

29. As per claim 11, Kimmel teaches a method of load balancing processors in a multiple processor system having a plurality of multi-processor modules, wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip

run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues, the method comprising the computer implemented steps of: comparing a first thread load of a first chip run queue of the plurality of chip run queues dedicated to a first multi-processor module of the plurality of multi-processor modules with a second thread load of a second chip run queue of a plurality of chip run queues dedicated to a second multi-processor module of the plurality of multi-processor modules; and reassigning a thread of the first chip run queue to the second chip run queue (Column 3, lines 5-7; Column 17, lines 46-60).

30. As per claim 12, Kimmel teaches wherein the step of comparing further comprises: determining a difference between the first thread load of the first chip run queue and the second thread load of the second chip run queue, reassigning the thread responsive to evaluating the difference as greater than a threshold (Column 17, lines 46-60).

Response to Arguments

31. Applicant's arguments with respect to claims 1-20 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to MENGYAO ZHE whose telephone number is (571)272-

6946. The examiner can normally be reached on Monday Through Friday, 7:30 - 5:00 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on 571-272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Meng-Ai An/
Supervisory Patent Examiner, Art Unit 2195